



INTEL® ADVISOR

VECTORIZATION ADVISOR

Get Faster Code Faster! Intel® Advisor

Vectorization Optimization

Have you:

- Recompiled for AVX2 with little gain
- Wondered where to vectorize?
- Recoded intrinsics for new arch.?
- Struggled with compiler reports?

Data Driven Vectorization:

New!

- What vectorization will pay off most?
- What's blocking vectorization? Why?
- Are my loops vector friendly?
- Will reorganizing data increase performance?
- Is it safe to just use pragma simd?

Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 54.44s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loop	Vector Issues	Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization?	Vectorized Loops
							Vecto... Efficiency
[loop at stl_algo.h:4740 i...		0.170s	0.170s		Scalar	non-vectorizable l ...	
[loop at loopstl.cpp:2449...	2 Ineffective peeled...	0.170s	0.170s	12; 4	Collapse	Collapse	AVX ~100%
[loop at loopstl.cpp:2...		0.150s	0.150s	12	Vectorized (B		AVX
[loop at loopstl.cpp:2...		0.020s	0.020s	4	Remainder		
[loop at loopstl.cpp:7900...		0.170s	0.170s	500	Scalar	vectorization possi...	
[loop at loopstl.cpp:35...	1 High vector regi...	0.160s	0.160s	12	Expand	Expand	AVX ~69%

"Intel® Advisor's Vectorization Advisor permitted me to focus my work where it really mattered. When you have only a limited amount of time to spend on optimization, it is invaluable."

Gilles Civario
Senior Software Architect
Irish Centre for High-End Computing

The Right Data At Your Fingertips

Get all the data you need for high impact vectorization



Filter by which loops are vectorized!

Trip Counts

What prevents vectorization?

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Trip Counts	Loop Type	Why No Vectorization?	Vectorized Loops		
							Vector...	Efficiency	Vector L...
[loop at stl_algo.h:4740 in std::tr...		0.170s	0.170s	12; 4	Scalar	non-vectorizable loop ins...			
[loop at loopstl.cpp:2449 in s234_]	2 Ineffective peeled/rem...	0.170s	0.170s	12	Collapse	Collapse	AVX	~100%	4
[loop at loopstl.cpp:2449 in s...		0.150s	0.150s	4	Vectorized (Body)		AVX		4
[loop at loopstl.cpp:2449 in s...		0.020s	0.020s	4	Remainder				
[loop at loopstl.cpp:7900 in vas_]		0.170s	0.170s	500	Scalar	vectorization possible but...			4
[loop at loopstl.cpp:3509 in s2...	1 High vector register ...	0.160s	0.160s	12	Expand	Expand	AVX	~69%	8
[loop at loopstl.cpp:3891 in s279_]	2 Ineffective peeled/rem...	0.150s	0.150s	125; 4	Expand	Expand	AVX	~96%	8
[loop at loopstl.cpp:6249 in s414_]		0.150s	0.150s	12	Expand	Expand	AVX	~100%	4
[loop at stl_numeric.h:247 in std...	1 Assumed dependency...	0.150s	0.150s	49	Scalar	vector dependence preve...			

Focus on hot loops

What vectorization issues do I have?

Which Vector instructions are being used?

How efficient is the code?

Get Faster Code Faster!

4 Steps to Efficient Vectorization

Intel® Advisor – Vectorization Advisor

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time	Compiler/Vectorization
			Loop Type Why No Vectorization?
[loop in runCForAllLambdaLoops]	0.094s	0.094s	Scalar vector dependence prevents vector...
[loop in runCForAllLambdaLoops]	0.140s	3.744s	Scalar inner loop was already vectorized
[loop in std::complex_base<double,struct_C_double_complex>::...	0.031s	0.031s	Vectorized (Body)
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations			
Peeled loop; loop stats were reordered			
[loop in std::basic_string<char,struct_std::char_traits<char>,class_std::allo...	0.000s	544.0...	Scalar nonstandard loop is not a vectoriza...
[loop in std::basic_string<char,struct_std::char_traits<char>,class_std::allo...	0.000s	544.0...	Scalar nonstandard loop is not a vectoriza...
[loop in std::num_put<char,class_std::ostreambuf_iterator<char,struct st...	0.000s	0.234s	Scalar nonstandard loop is not a vectoriza...

2. Guidance: detect problem and recommend how to fix it

Issue: Peeled/Remainder loop(s) present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials](#), [Utilizing Full Vectors](#).

Recommendation: Align memory access
 Projected maximum performance gain: High
 Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary:

```
float *array;
array = (float *)_mm_malloc(ARRAY_SIZE*sizeof(float), 32);

// Somewhere else
_assume_aligned(array, 32);
// Use array in loop
```

3. Loop-Carried Dependency Analysis

Problems and Messages						
ID	Type	Site Name	Sources	Modules	State	
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem	
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	✗ New	
P7	Write after read dependency	site2	dqtest2.cpp, idl.e	dqtest2	✗ New	

4. Memory Access Patterns Analysis

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_203	runCRawLoops	runCRawLoops.cox1063	RAW1	No information available	No information available
loop_site_139	runCRawLoops	runCRawLoops.cox622	No information available	39% / 36% / 25%	Mixed strides
loop_site_160	runCRawLoops	runCRawLoops.cox925	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns	Correctness Report				
ID	Stride	Type	Source	Modules	Alignment
P22	0; 0; 1	Unit stride	runCRawLoops.cox637	lcals.exe	
P23	0; 0	Unit stride	runCRawLoops.cox638	lcals.exe	
P30	-1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801	Variable stride	runCRawLoops.cox628	lcals.exe	

```

635 j2 = ( j2 + 64 - 1 ) ;
636 p[4p][0] += y[12+32];
637 p[4p][1] += z[32+32];
638 i2 += e[12+32];
639 j2 += f[32+32];
    
```

```

626 i1 = 64 - 1;
627 j1 = 64 - 1;
628 p[4p][2] += b[j1][11];
    
```

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
 *Other names and brands may be claimed as the property of others.



1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time	Compiler/Vectorization	
			Loop Type	Why No Vectorization?
[loop in runCforallLambdaLoops]	0.094s	0.094s	Scalar	vector dependence prevents vector...
[loop in runCforallLambdaLoops]	0.140s	3.744s	Scalar	inner loop was already vectorized
▼ [loop in std::complex_base<double,struct _C_double_complex>::i...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations				
Peeled loop; loop stmts were reordered				
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	0.234s	Scalar	nonstandard loop is not a vectoriza...

Is Most Execution in the Fast Part of the Vector?

Intel Advisor shows you

Where should I add vectorization and/or threading parallelism?

Summary | Survey Report | Refinement Reports | Annotation Report | Suitability Report

Elapsed time: 8,52s | Vectorized | Not Vectorized | FILTER: All Modules | All Sources

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?
[loop at fractal.cpp:179 in <lambda1>::op...	4 High vector ...	0,013s	12,020s	Collapse	Collapse
[loop at fractal.cpp:179 in <lambda1>::o...	4 Serialized use ...	0,013s	11,281s	Vectorized (Body)	
[loop at fractal.cpp:179 in <lambda1>::o...	2 Data type co...	0,000s	0,163s	Peeled	
[loop at fractal.cpp:179 in <lambda1>::o...	2 Data type co...	0,000s	0,576s	Remainder	
[loop at fractal.cpp:177 in <lambda1>::oper...	2 Data type co...	0,010s	12,030s	Scalar	

File: fractal.cpp:164 <lambda1>::operator()

Line	Source	Total Time	%	Loop Time	%
163	for (int x = x0; x < x1; ++x) { [loop at fractal.cpp:163 in <lambda1>::operator()] Scalar Loop. Not vectorized: outer loop was not auto-vectorized: consider us... No loop transformations were applied			10.822s	
164	for (int y = y0; y < y1; ++y) { [loop at fractal.cpp:164 in <lambda1>::operator()] Scalar Loop. Not vectorized: vectorization possible but seems inefficient. Us... Loop was unrolled by 2			10.822s	
165	fractal_data_array[x - x0][y - y0] = calc_one_pixel(x, y, t	10.822s			
166	}				
167	}				
168	for (int y = y0, y_temp = 0; y < y1; ++y, ++y_temp) {				
169	area.set_pos(0, y - y0);				
170	for (int x = x0, x_temp = 0; x < x1; ++x, ++x_temp) {				
171	area.put_pixel(fractal_data_array[x_temp][y_temp]);				
172	}				
173	}	0.196s			

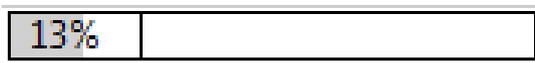
Optimization Notice

Vector Efficiency: All The Data In One Place

My “performance thermometer”

Elapsed time: 8,01s

Loops	Vecto...	Efficiency ▲	Estimated Gain	Vect...	Co	Traits	Vector Widths	Self Time
[loop at lbpSUB.cpp:1280 in fPropagationS...	AVX	13%	0,53	4	0,53	Blends; Extracts; Inserts; Shuffles	128/256	2,312s
[loop at lbpGET.cpp:152 in fGetFracSite]	AVX	30%	2,38	8	2,34	Blends; Inserts; Masked Stores	128/256	0,030s
[loop at lbpGET.cpp:42 in fGetOneMassSite]	AVX	36%	2,86	8	2,79		256	0,100s
[loop at lbpGET.cpp:78 in fGetTotMassSite]	AVX	36%	2,86	8	2,79		256	0,010s
[loop at lbpGET.cpp:334 in fGetOneDirecSp ...]	AVX	38%	3,05	8	2,97	Type Conversions	128/256	0,011s
[loop at lbpBGK.cpp:840 in fCollisionBGK]	AVX	100%	2,05	2	2,05		128	0,080s



Achieved Efficiency

Original (scalar) code efficiency. Corresponds to 1x speed-up.

Upper bound: 100% efficiency 4x gain (VL=4)

- **Auto-vectorization:** affected <3% of code
 - With moderate speed-ups
- First attempt to **simply put #pragma simd:**
 - Introduced slow-down
- Look at Vector Issues and **Traits** to find out **why**
 - All kinds of “memory manipulations”
 - Usually an indication of “bad” access pattern

Survey: Find out if your code is “under vectorized” and why

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization	
			Loop Type	Why No Vectorization?
loop in runCForsAllLambdaLoop]	0.094s	0.094s	Scalar	vector dependence prevents vector...
loop in runCForsAllLambdaLoop]	0.140s	3.744s	Scalar	inner loop was already vectorized
loop in std::complex_base::double_struct::C_double_complex::...	0.031s	0.031s	Vectorized (Body)	
Vectorized SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations				
Peeled loop; loop starts were reordere...				
loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
loop in std::basic_string<char,struct std::char_traits<char>,class std::allo...	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriza...
loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...	0.000s	0.234s	Scalar	nonstandard loop is not a vectoriza...

2. Guidance: detect problem and recommend how to fix it



Issue: Peeled/Remainder loop(s) present



All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [Vector Essentials, Utilizing Full Vectors...](#)

Recommendation: Align memory access

Projected maximum performance gain: High
Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary:

```
float *array;
array = (float *)_mm_malloc(ARRAY_SIZE*sizeof(float), 32);

// Somewhere else
    _assume_aligned(array, 32);
// Use array in loop
```

Get Specific Advice For Improving Vectorization

Intel® Advisor – Vectorization Advisor

The screenshot shows the Intel Advisor XE 2016 interface. At the top, there's a title bar with the text "Where should I add vectorization and/or threading parallelism?". Below that, there are navigation tabs: Summary, Survey Report, Refinement Reports, Annotation Report, and Suitability Report. A search bar and filter options are visible. The main area displays a table of function call sites and loops. One row is highlighted in blue, indicating an issue: "[loop at fractal.cpp:179 in <lambda1>::op ...]". A blue callout box points to this row with the text "Click to see recommendation". Below the table, there are tabs for "Top Down", "Source", "Loop Assembly", "Assistance", "Recommendations", and "Compiler Diagnostic Details". The "Recommendations" tab is active, showing a detailed message: "Issue: Ineffective peeled/remainder loop(s) present". The message explains that some source loop iterations are not executing in the loop body and suggests moving them to the loop body. A table of directives is provided: ICL/ICC/ICPC Directive (#pragma nounroll, #pragma unroll) and IFORT Directive (IDIRS NOUNROLL, IDIRS UNROLL). A blue callout box points to this section with the text "Advisor shows hints to move iterations to vector body." Below the table, there is a "Read More:" section with a link to the Intel C++ Compiler 15.0 User and Reference Guide.

Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Loop Type	Why No Vectorization?	Vectorized Loops
						Vecto... Estim... Vector Len
[loop at market ...]			11,460s	Scalar		
[loop at arena.cpp:88 in tbb::tbb::]		0,000s	11,460s	Scalar		
[loop at fractal.cpp:179 in <lambda1>::op ...]	Ineffective ...	0,000s	2,022s	Collapse	Collapse	
[loop at fractal.cpp:179 in <lambda1>::o ...]	Data type co ...	0,000s	2,022s	Remainder		

Issue: Ineffective peeled/remainder loop(s) present

All or some [source loop](#) iterations are not executing in the [loop body](#). Improve performance by moving source loop iterations from [peeled/remainder](#) loops to the loop body.

Disable unrolling
The [trip count](#) after loop unrolling is too small compared to [unroll factor](#) using a [directive](#).

ICL/ICC/ICPC Directive	IFORT Directive
#pragma nounroll	IDIRS NOUNROLL
#pragma unroll	IDIRS UNROLL

Read More:

- [User and Reference Guide for the Intel C++ Compiler 15.0 > Compiler Reference > Pragmas > Intel-specific Pragma Reference > unroll/nounroll.](#)

Critical Data Made Easy

Loop Trip Counts

Knowing the time spent in a loop is not enough!

Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

Summary Survey Report Refinement Reports Annotation Reports Suitability Report

Program time: 12.82s Vectorized Not Vectorized Filter: All Modules All Sources

Function Call Sites and Loops	Self Time	Total Time	Fire	Lightbulb	Trip Counts				Compiler Vectorization	
					Median	Min	Max	Call Count	Loop Type	Why No Vectorization
[loop at Multiply.c:53 in matvec]	11.898s	11.898s		1					Collapse	Collapse
↳ [loop at Multiply.c:53 in matvec]	11.851s	11.851s		1	101	101	101	12000000	Vectorized (Body)	vector dependence p
↳ [loop at Multiply.c:53 in matvec]	0.047s	0.047s			3	3	3	1000000	Vectorized (Body)	
↳ [loop at Multiply.c:53 in matvec]	0.413s	0.413s			101	101	101	2000000	Scalar	
↳ [loop at Multiply.c:45 in matvec]	0.109s	12.373s		1					Expand	Expand
↳ [loop at Driver.c:146 in main]	0.016s	12.483s		1	1000000	1000000	1000000	1	Scalar	vector dependence p

1.1 Find Trip Counts

Find how many iterations are executed.

▶ 📁

Command Line

Check actual trip counts

Loop is iterating 101 times but called > million times

Since the loop is called so many times it would be a big win if we can get it to vectorize.

Optimization Notice

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops	Self Time	Total Time	Compiler Vectorization	
			Loop Type	Why No Vectorization?
[[loop in runCforallLambdaLoop]]	0.094	0.094	Scalar	vector dependence prevents vector...
[[loop in runCforallLambdaLoop]]	0.140	3.744	Scalar	inner loop was already vectorized
[[loop in std::complex_base<double,struct C_double,complex2...]]	0.0314	0.0314	Vectorized (Hoist)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations Peeled loop; loop stats were reordered				
[[loop in std::basic_string_char<struct std::char_traits<char>,class std::allo...]]	0.000	544.0	Scalar	nonstandard loop is not a vectoriz...
[[loop in std::basic_string_char<struct std::char_traits<char>,class std::allo...]]	0.000	544.0	Scalar	nonstandard loop is not a vectoriz...
[[loop in std::num_put<char,class std::ostreambuf_iterator<char,struct st...]]	0.000	0.234	Scalar	nonstandard loop is not a vectoriz...

2. Guidance: detect problem and recommend how to fix it

Issue: Peeled/Remainder loop(s) present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [vector Essentials: Utilizing Full Vectors](#).

Recommendation: Align memory access

Projected maximum performance gain: High
Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary.

```
float *array;
array = (float *)_m_malloc(ARRAY_SIZE*sizeof(float), 32);

// Somewhere else
_assume_aligned(array, 32);
// Use array in loop
```

3. Loop-Carried Dependency Analysis

Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P7	Write after read dependency	site2	dqtest2.cpp; idle.h	dqtest2	🔴 New

Is It Safe to Vectorize?

Loop-carried dependencies analysis verifies correctness

Where should I add vectorization and/or threading parallelism? Intel Advisor XE 2016

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Program time: 12.82s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops	Self Time	Total Time	🔥	💡	Trip Counts	Compiler Vectorization	
						Loop Type	Why No Vectorization?
↳ [loop at Multiply.c:53 in matvec]	0.047s	0.047s	<input type="checkbox"/>		3	Vectorized (Body)	
↳ [loop at Multiply.c:53 in matvec]	0.413s	0.413s	<input type="checkbox"/>		101	Scalar	
↳ [loop at Multiply.c:45 in matvec]	0.109s	12.373s	<input type="checkbox"/>	💡 1		Collapse	Collapse
↳ [loop at Multiply.c:45 in matvec]	0.078s	11.930s	<input type="checkbox"/>		12	Vectorized (Body)	
↳ [loop at Multiply.c:45 in matvec]	0.031s	0.444s	<input type="checkbox"/>		2	Remainder	
↳ [loop at Driver.c:146 in main]	0.016s	12.483s	<input checked="" type="checkbox"/>	💡 1	1000000	Scalar	vector dependence prevents vectoriza...

2.1 Check Correctness

Identify and explore loop-carried dependencies for marked loops. Fix the reported problems.

▶ 📄

Command Line

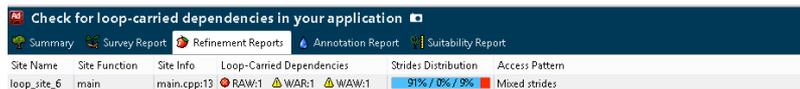
Select loop for Correct Analysis and press play!

Vector Dependence prevents Vectorization!

Optimization Notice

Correctness – Is It Safe to Vectorize?

Loop-carried dependencies analysis



Detected dependencies

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	loop_site_6	main.cpp	test_1.exe	✓ Not a problem
P3	Read after write dependency	loop_site_6	critexe.c; main.cpp	test_1.exe	New
P4	Write after write dependency	loop_site_6	critexe.c; main.cpp	test_1.exe	New
P5	Write after read dependency	loop_site_6	critexe.c; main.cpp	test_1.exe	New

ID	Description	Source	Function	Module	State
X17	Read	main.cpp:22	main	test_1.exe	New
20 k += a[9];					
21 k -= a[8];					
22 k -= a[7];					
23 k += a[6];					
24 k -= a[5];					
X18	Read	main.cpp:23	main	test_1.exe	New
21 k -= a[8];					
22 k -= a[7];					
23 k += a[6];					

Source lines with Read and Write accesses detected

Received recommendations to force vectorization of a loop:

1. Mark-up loop and check for REAL dependencies
2. Explore dependencies with code snippets

In this example 3 dependencies were detected:

- RAW – Read After Write
- WAR – Write After Read
- WAW – Write After Write

This is NOT a good candidate to force vectorization!

1. Compiler diagnostics + Performance Data + SIMD efficiency information

Function Call Sites and Loops*	Self Time	Total Time	Compiler Vectorization	
			Loop Type	Why No Vectorization?
ii [loop in runCforallLambdaLoop]	0.094s	0.094s	Scalar	vector dependence prevents vector...
iii [loop in runCforallLambdaLoop]	0.140s	3.744s	Scalar	inner loop was already vectorized
iv [loop in std::complex_base_double_struct_C_double_complex...]	0.031s	0.031s	Vectorized (Hoist)	
Vectorized SSE; SSE2 loop processing Float32; Float64 data type(s) having Divisions; Square Roots operations Peeled loop; loop stats were reordered				
ii [loop in std::basic_string_char_struct_std_char_traits_cchar_class_std::allo...]	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriz...
iii [loop in std::basic_string_cchar_struct_std_char_traits_cchar_class_std::allo...]	0.000s	544.0...	Scalar	nonstandard loop is not a vectoriz...
ii [loop in std::num_put_cchar_class_std::ostreambuf_iterator_cchar_struct.st...]	0.000s	0.234s	Scalar	nonstandard loop is not a vectoriz...

2. Guidance: detect problem and recommend how to fix it

Issue: Peeled/Remainder loop(s) present

All or some source loop iterations are not executing in the kernel loop. Improve performance by moving source loop iterations from peeled/remainder loops to the kernel loop. Read more at [vector Essentials: Utilizing Full Vectors...](#)

Recommendation: Align memory access

Projected maximum performance gain: High
Projection confidence: Medium

The compiler created a peeled loop because one of the memory accesses in the source loop does not start at a data boundary. Align the memory access and tell the compiler your memory access is aligned. This example aligns memory using a 32-byte boundary.

```
float *array;
array = (float *)_m_malloc(ARRAY_SIZE*sizeof(float), 32);

// Somewhere else
_assume_aligned(array, 32);
// Use array in loop
```

3. Loop-Carried Dependency Analysis

Problems and Messages

ID	Type	Site Name	Sources	Modules	State
P1	Parallel site information	site2	dqtest2.cpp	dqtest2	✓ Not a problem
P2	Read after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P3	Read after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P4	Write after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P5	Write after write dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P6	Write after read dependency	site2	dqtest2.cpp	dqtest2	🔴 New
P7	Write after read dependency	site2	dqtest2.cpp, idle.h	dqtest2	🔴 New

4. Memory Access Patterns Analysis

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_203	runCRawLoops	runCRawLoops.cxx:1063	RAW-1	No information available	No information available
loop_site_139	runCRawLoops	runCRawLoops.cxx:622	No information available	39% / 36% / 25%	Mixed strides
loop_site_160	runCRawLoops	runCRawLoops.cxx:925	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns	Correctness Report				
ID	Stride	Type	Source	Modules	Alignment
P22	0; 0; 1	Unit stride	runCRawLoops.cxx:637	lcal.exe	
<pre> 635 j2 = (j2 + 64 - 1) ; 636 p[ip][0] += y[i2+32]; 637 p[ip][1] += z[j2+32]; 638 i2 += e[i2+32]; 639 j2 += f[j2+32]; </pre>					
P23	0; 0	Unit stride	runCRawLoops.cxx:638	lcal.exe	
<pre> 626 i1 = 64 - 1; 627 j1 = 64 - 1; 628 p[ip][2] += b[j1][i1]; </pre>					
P30	-1575; -63; -26; -25; -1; 0; 1; 25; 26; 63; 2164801	Variable stride	runCRawLoops.cxx:628	lcal.exe	

Improve Vectorization

Memory Access pattern analysis

Where should I add vectorization and/or threading parallelism?

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Elapsed time: 8,52s Vectorized Not Vectorized FILTER: All Modules All Sources

Function Call Sites and Loops				Loop Type	Why No Vectorization?	
[loop at fractal.cpp:179 in <lambda1>::op ...]	<input type="checkbox"/>	4 M ...	0,013s I	12,020s	Collapse	Collapse
[loop at fractal.cpp:179 in <lambda1>::o ...]	<input checked="" type="checkbox"/>	Serialized use ...	0,013s I	11,281s	Vectorized (Body)	
[loop at fractal.cpp:179 in <lambda1>::o ...]	<input checked="" type="checkbox"/>	Data type co ...	0,000s I	0,163s I	Peeled	
[loop at fractal.cpp:179 in <lambda1>::o ...]	<input checked="" type="checkbox"/>	Data type co ...	0,000s I	0,576s I	Remainder	
[loop at fractal.cpp:177 in <lambda1>::oper ...]	<input type="checkbox"/>	Data type co ...	0,010s I	12,030s	Scalar	

Select loops of interest

2.2 Check Memory Access Patterns
Identify and explore complex memory accesses for marked loops. Fix the reported problems.

Command Line

Run Memory Access Patterns analysis, just to check how memory is used in the loop and the called function

Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.



Find vector optimization opportunities

Memory Access pattern analysis

Stride distribution

Check memory access patterns in your application Intel Advisor XE 2016

Summary Survey Report Refinement Reports Annotation Report Suitability Report

Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_79	operator()	fractal.cpp:179	No information available	100% / < 1,0000% / ...	Mixed strides
loop_site_93	operator()	fractal.cpp:179	No information available	100% / 0% / 0%	All unit strides
loop_site_94	operator()	fractal.cpp:179	No information available	100% / 0% / 0%	All unit strides

Memory Access Patterns Report

ID	Stride	Type	File	File
P18	0	Unit stride		
P21	0	Unit stride	fractal.cpp:66	fractal.exe
P24	0	Unit stride	fractal.cpp:68	fractal.exe
P27	0	Unit stride	fractal.cpp:69	fractal.exe
P30	0	Unit stride	fractal.cpp:74	fractal.exe

```
64 color_t color;
65
66 fx0 = x0 - size_x / 2.0f;
67 fy0 = y0 - size_y / 2.0f;
68 fx0 = fx0 / magn + cx;
```

```
66 fx0 = x0 - size_x / 2.0f;
67 fy0 = y0 - size_y / 2.0f;
68 fx0 = fx0 / magn + cx;
69 fy0 = fy0 / magn + cy;
70
```

All memory accesses are uniform, with zero unit stride, so the same data is read in each iteration

We can therefore declare this function using the omp syntax: `pragma omp declare simd uniform(x0)`

Optimization Notice

Quickly Find Loops with Non-optimal Stride

Memory Access pattern analysis

- Quickly identify loops that are good, bad or mixed.
- Unit stride memory accesses are preferable.
- Find unaligned data

Check memory access patterns in your application Intel Advisor XE 2016

Summary Survey Report Refinement Reports MAP Source: fractal.cpp Annotation Report Suitability Report

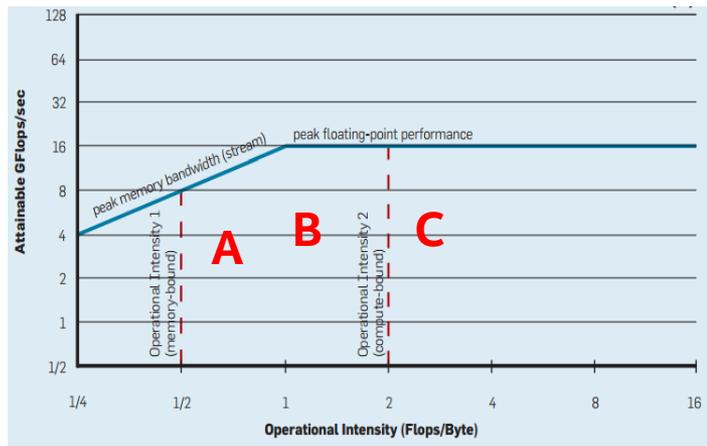
Site Name	Site Function	Site Info	Loop-Carried Dependencies	Strides Distribution	Access Pattern
loop_site_54	operator()	fractal.cpp:164	✔ No dependencies found	No information available	No information available
loop_site_129	operator()	fractal.cpp:164	No information available	100% / 0% / 0%	All unit strides

ID	Stride	Type	Source	Modules	Alignment
P1		Parallel site information	fractal.cpp:164	fractal.exe	
P3	0	Unit stride	fractal.cpp:100	fractal.exe	
<pre>98 } 99 #endif 100 int b = (int)(256 * mu); 101 int g = (b / 8); 102 int r = (g / 16);</pre>					
P4	0	Unit stride	fractal.cpp:164	fractal.exe	
<pre>162 163 for (int x = x0; x < x1; ++x) { 164 for (int y = y0; y < y1; ++y) { 165 fractal_data_array[x - x0][y - y0] = calc_one_pixel(x, y, tmp_max_iterations, tmp_size_x, tmp_si 166 }</pre>					
P5	0	Unit stride	fractal.cpp:164	fractal.exe	
P6	0; 1	Unit stride	fractal.cpp:165	fractal.exe	
P7	0; 1	Unit stride	fractal.cpp:165	fractal.exe	
P8	0	Unit stride	fractal.cpp:60	fractal.exe	
P9	0	Unit stride	fractal.cpp:66	fractal.exe	

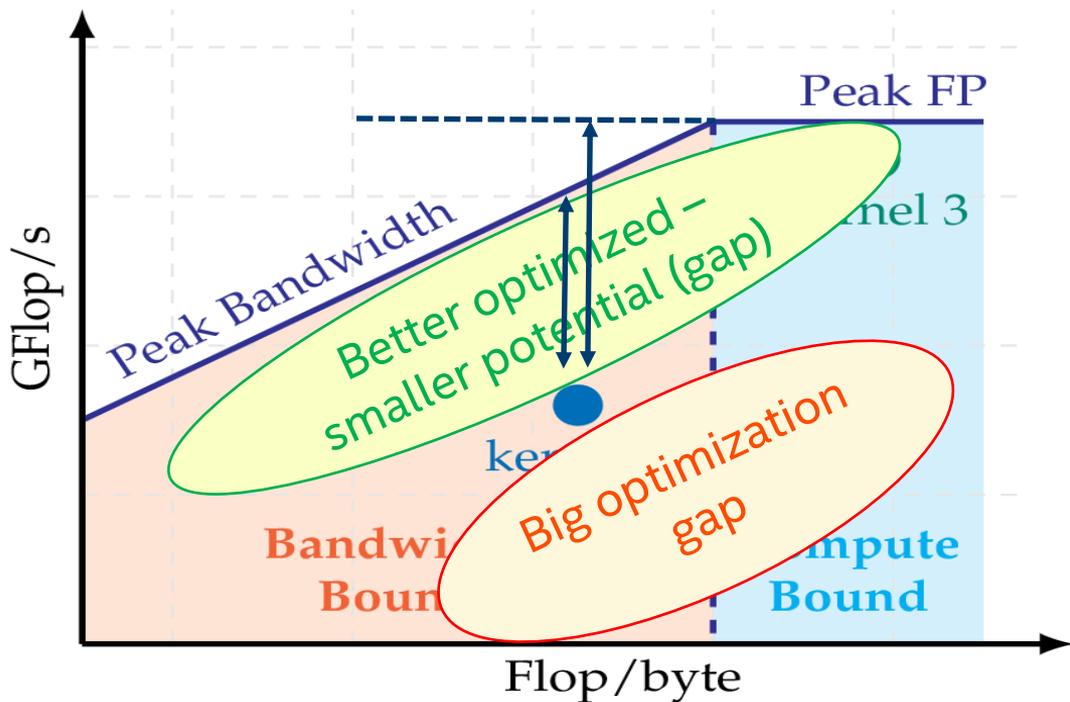
Optimization Notice

NEW ROOFLINE ANALYSIS

Roofline model: Am I bound by VPU/CPU or by Memory?

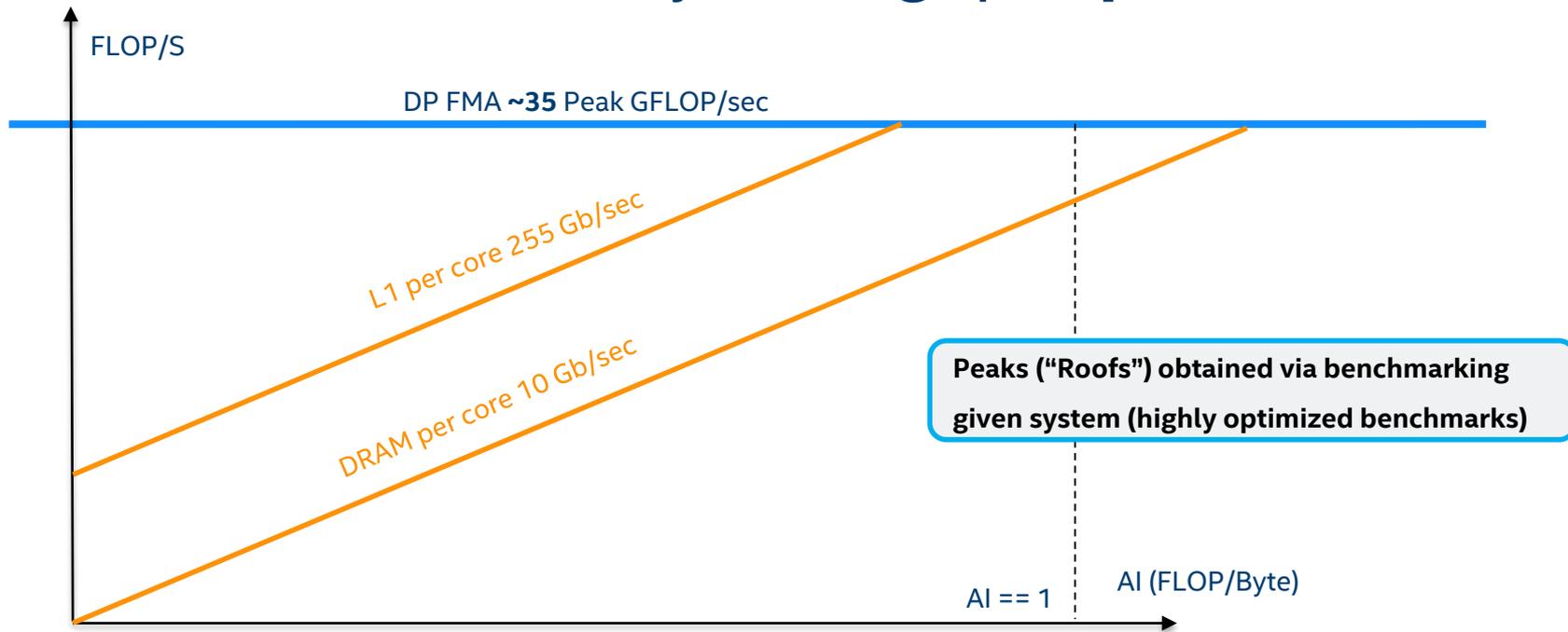


What makes loops **A**, **B**, **C** different?



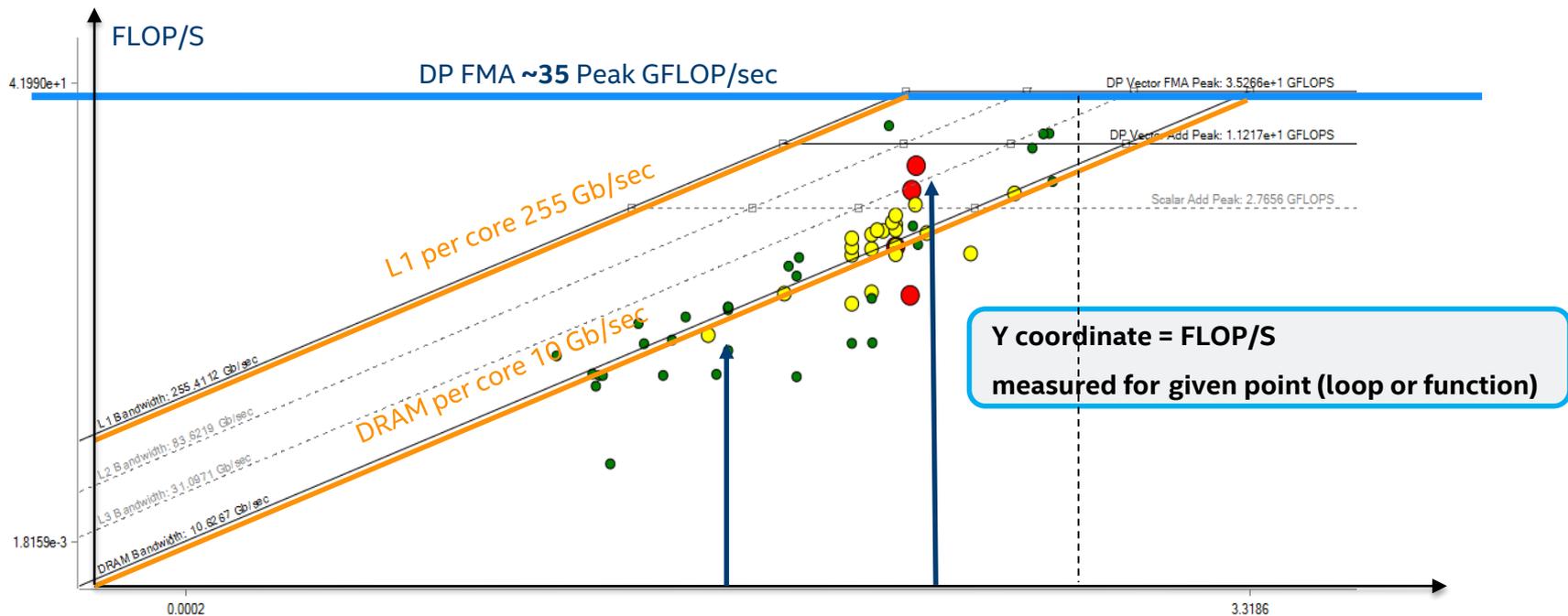
Roofline ingredient #1

a. FLOPS and b. Memory throughput peaks



Roofline ingredient #2

Axis Y: FLOP/S data



Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

Intel Confidential



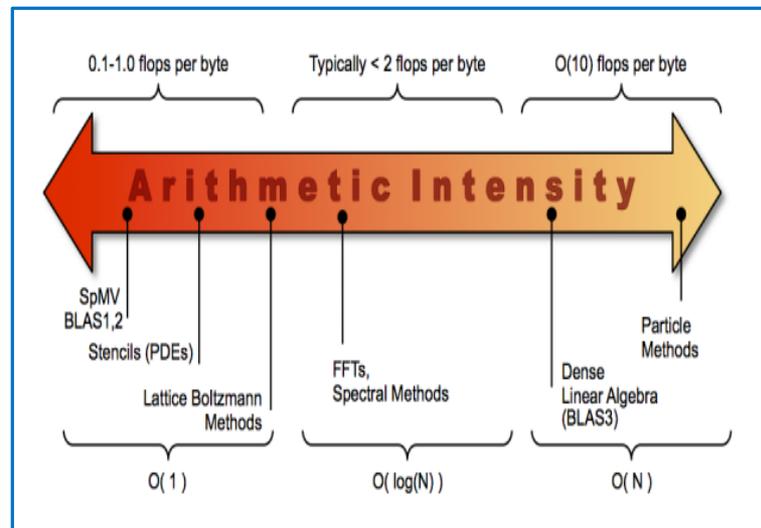
Roofline ingredient #3:

Axis X: Arithmetic Intensity (AI)

Putting

- memory utilization/demand
 - CPU utilization
- altogether

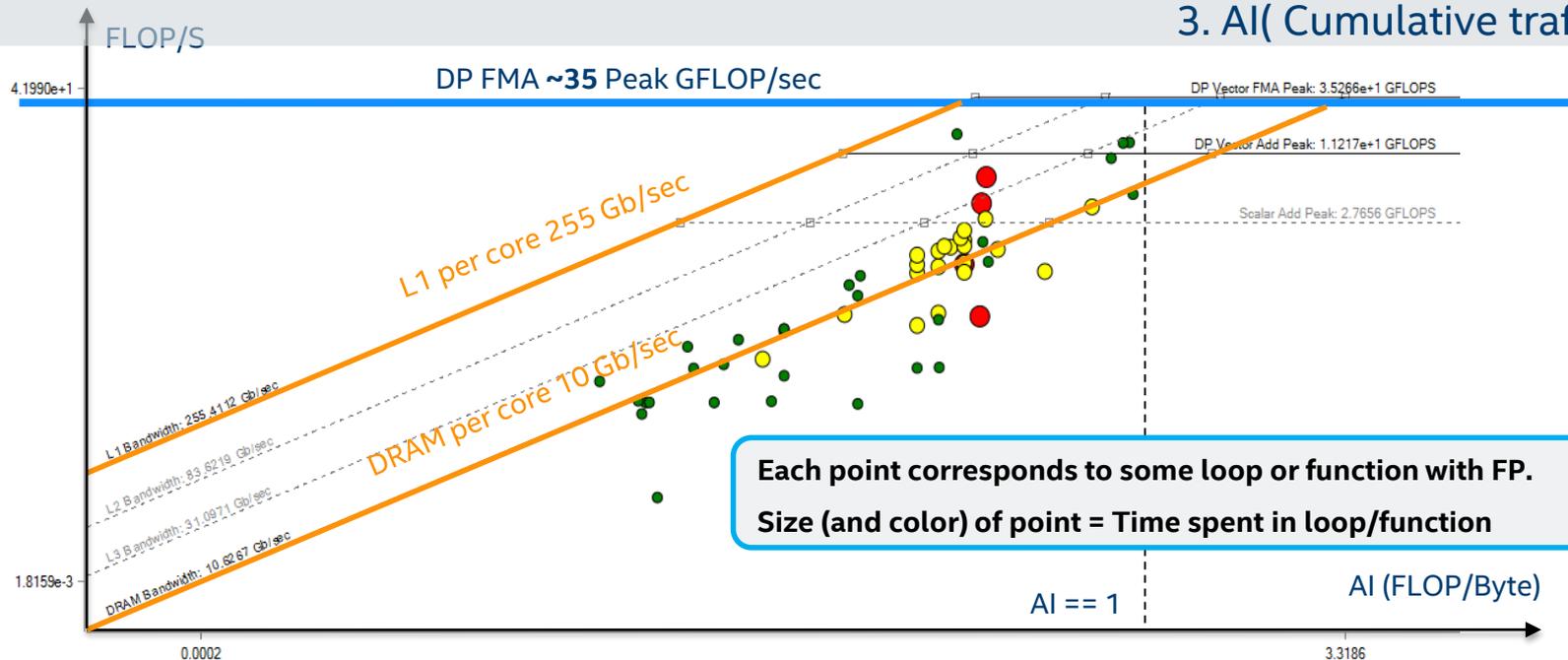
$$AI = \# \text{ FLOP} / \# \text{ BYTE}$$



What makes loops
A, B, C different?

3 ingredients => Intel Advisor Roofline automation

1. Roofs (benchmark-based)
2. FLOP/S (AVX-512 mask aware)
3. AI (Cumulative traffic)



Optimization Notice

Copyright © 2016, Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.

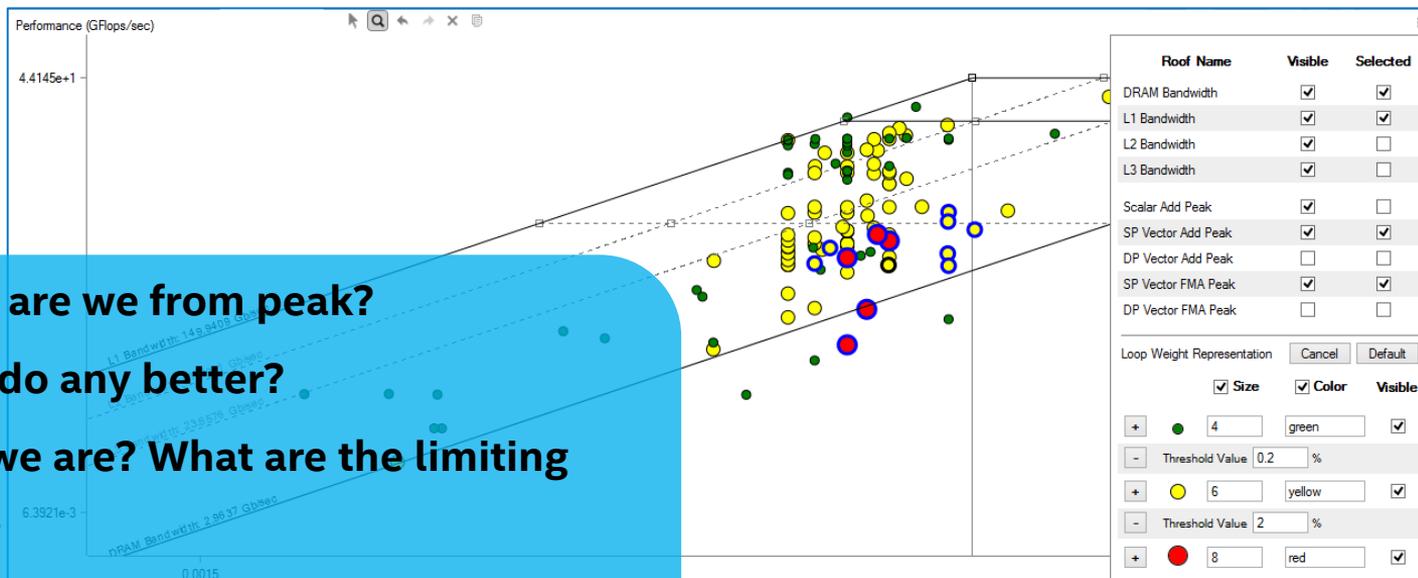
Intel Confidential



Roofline Automation in Intel Advisor 2017+ answer the questions

1. How far are we from peak?
2. Can we do any better?
3. Where we are? What are the limiting factors?

- Memory subsystem?
- Lack of CPU/Vectorization/Threading?
- Both? Anything else?



- Interactive mapping to source and performance profile
- Synergy between Vector Advisor and Roofline: FMA example
- Customizable chart

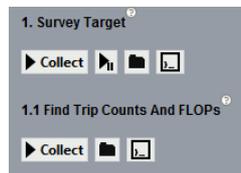
FLOP/s data measurement

1. Seconds are given by Survey run
2. #FLOP is currently given by “Trip Counts” run
 - Additionally provides cumulative memory traffic and (AVX-512) Mask Register Utilization profile

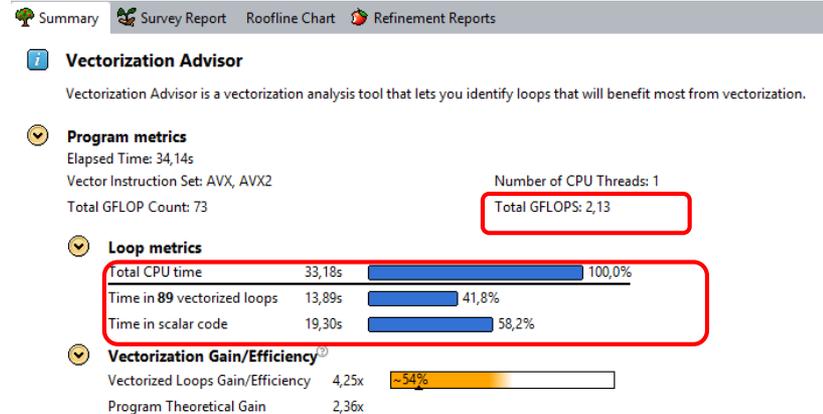
Works from Nehalem to KNL, technology mostly invariant to target platform

- Do not depend on PMU capabilities
- Good FLOPS/Mask PMU doesn't exist for KNL

Currently mapped to loops, functions, workload



FLOPs, Masks, Trip Counts					
Median	GFLOPs/s	Arithmetic Intensity	Mask Utiliz...	GBytes/s	GFLOP
19	2,456	0.125		19.6498	3.94488
4; 3	2,351	0.125	63,29%	18.8111	0.36693
19	2,136	0.0795455		26.8513	2.50206
19	1,910	0.0681818		28.011	1.07231
3	1,774	0.0833333		21.2898	0.11287
4	1,192	0.0666667		17.8726	0.1505
19	0,911	0.0681818		13.3635	0.0285



Cache-Aware vs. Classic Roofline

$$AI = \# \text{ FLOP} / \# \text{ BYTE}$$

- **AI_DRAM** (often referred to as Operational Intensity)

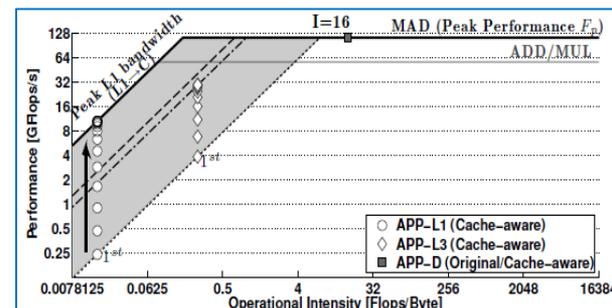
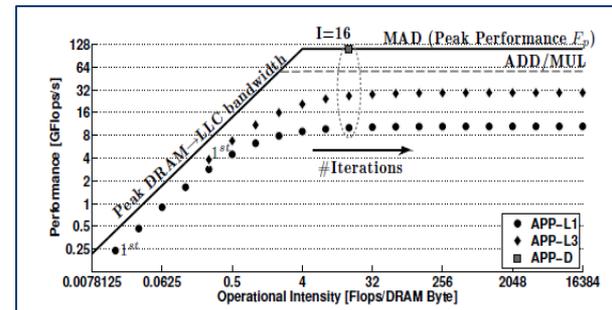
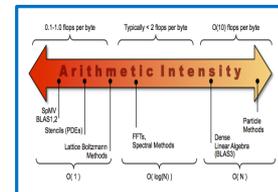
$$= \# \text{ FLOP} / \# \text{ BYTES (CPU \& Cache} \Leftrightarrow \text{DRAM)}$$

- “DRAM traffic”-based
- Variable for the same code/platform (varies with dataset size/trip count)
- Can be measured relative to different memory hierarchy levels – cache level, HBM, DRAM

- **AI_CARM**

$$= \# \text{ FLOP} / \# \text{ BYTES (CPU} \Leftrightarrow \text{Memory Sub-system)}$$

- “Algorithmic”, “Cumulative (L1+L2+LLC+DRAM) traffic-based”
- Invariant for the given code on given platform
- Typically $AI_CARM < AI_DRAM$



Acknowledgments/References

Classic Roofline formulated by Williams, Waterman, Patterson, (Berkeley)

<http://www.eecs.berkeley.edu/~waterman/papers/roofline.pdf>

“Cache-aware Roofline model: Upgrading the loft” (Ilic, Pratas, Sousa, INESC-ID/IST, Thec Uni of Lisbon)

<http://www.inesc-id.pt/ficheiros/publicacoes/9068.pdf>

Done by Intel Advisor and PathFinding Teams,
Roman Belenov, Igor Kaleturin, Julia Fedorova, Zakhar Matveev

in collaboration with Philippe Thierry and his colleagues.

Some implementation aspects were inspired by Intel SDE and Hugh Caffey M

**To Register for Advisor “Roofline” Alpha Evaluation:
Send request to vector_advisor@intel.com**

Intel® Parallel Studio XE

Faster code faster!

Intel Advisor is part of Intel Parallel Studio XE:

Vectorizing Compiler

Squeeze all the performance out of the latest instruction set

Threaded Performance Libraries

Pre-vectorized, pre-threaded, pre-optimized

High Level Parallel Models

Productive solutions for thread, process & vector parallelism

Parallel Performance Profilers

Quickly discover bottlenecks and tune for high performance

Thread Debugger

Find and debug non-deterministic threading errors

Vectorization Optimization and Thread Prototyping

Data driven design tools help you vectorize & thread effectively



Download Today

Google:

“Intel Parallel Studio”

Or go directly to:

<https://software.intel.com/en-us/articles/intel-parallel-studio>

Additional Resources

All links start with: <https://software.intel.com/>

Vectorization Guide: <https://software.intel.com/articles/a-guide-to-auto-vectorization-with-intel-c-compilers/>

Explicit Vector Programming in Fortran:

<https://software.intel.com/articles/explicit-vector-programming-in-fortran>

Optimization Reports: <https://software.intel.com/videos/getting-the-most-out-of-the-intel-compiler-with-new-optimization-reports>

Beta Registration & Download: <https://software.intel.com/en-us/articles/intel-parallel-studio-xe-2016-beta>

For Intel® Xeon Phi™ coprocessors, but also applicable:

<https://software.intel.com/en-us/articles/vectorization-essential>

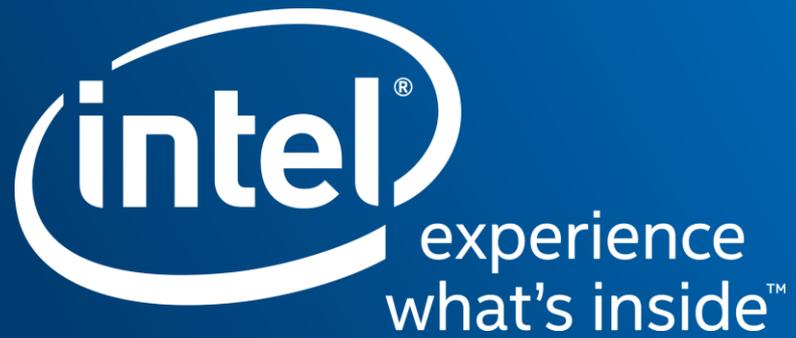
<https://software.intel.com/en-us/articles/fortran-array-data-and-arguments-and-vectorization>

Intel® Composer XE User and Reference Guides:

https://software.intel.com/compiler_15.0_ug_c

https://software.intel.com/compiler_15.0_ug_f

Compiler User Forums: <http://software.intel.com/forums>



Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

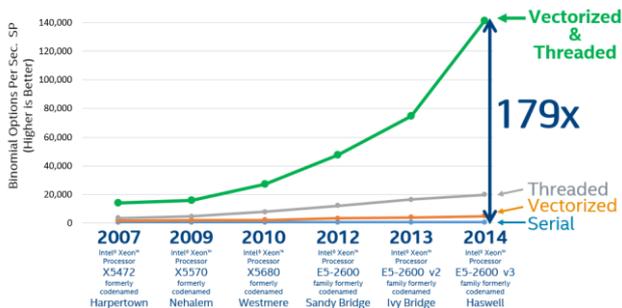
Copyright © 2015, Intel Corporation. All rights reserved. Intel, Pentium, Xeon, Xeon Phi, Core, VTune, Cilk, and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Configurations for Binomial Options SP



Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804

Performance measured in Intel Labs by Intel employees

Platform Hardware and Software Configuration

Platform	Unscaled Core Frequency	Cores/Socket	Num Sockets	L1 Data Cache	L1 I Cache	L2 Cache	L3 Cache	Memory	Memory Frequency	Memory Access	H/W Prefetchers Enabled	HT Enabled	Turbo Enabled	C States	O/S Name	Operating System	Compiler Version
Intel® Xeon™ 5472 Processor	3.0 GHZ	4	2	32K	32K	12 MB	None	32 GB	800 MHZ	UMA	Y	N	N	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ X5570 Processor	2.93 GHZ	4	2	32K	32K	256K	8 MB	48 GB	1333 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ X5680 Processor	3.33 GHZ	6	2	32K	32K	256K	12 MB	48 MB	1333 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2690 Processor	2.9 GHZ	8	2	32K	32K	256K	20 MB	64 GB	1600 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y	Disable	Fedora 20	3.11.10-301.fc20	icc version 14.0.1
Intel® Xeon™ E5 2697v2 Processor	2.7 GHZ	12	2	32K	32K	256K	30 MB	64 GB	1867 MHZ	NUMA	Y	Y	Y				